



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/870,621	05/31/2001	Scott J. Broussard	AUS920010263US1	1785
35617 7590 03/07/2007 DAFFER MCDANIEL LLP P.O. BOX 684908 AUSTIN, TX 78768			EXAMINER BONSHOCK, DENNIS G	
			ART UNIT	PAPER NUMBER
			2173	
SHORTENED STATUTORY PERIOD OF RESPONSE		MAIL DATE	DELIVERY MODE	
2 MONTHS		03/07/2007	PAPER	

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.



UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Application Number: 09/870,621
Filing Date: May 31, 2001
Appellant(s): BROUSSARD, SCOTT J.

MAILED

MAR 07 2007

Technology Center 2100

Kevin L. Daffer (reg. 34,146)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 11-15-2006.

(1) *Real Party in Interest*

A statement identifying the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The following are the related appeals, interferences, and judicial proceedings known to the examiner which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal:

09/870,613

09/870,615

09/870,620

09/870,621

09/870,622

09/870,624

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5,327,529

Fults

7-1994

Sun Microsystems, Java Platform 1.2 Beta 4 API Specification: Class JpasswordField and Class JPasswordField, 1993-1998, page 1, hereinafter Java

Sun Microsystems, The Swing Connection, 2/98, volumn 3, no.4, swing version 1.0, hereinafter Java (specifically IS).

WinZip Computing Inc., WinZip 8.0, 1991-2000, attached pages, hereinafter WinZip

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

5. Claims 1-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over WinZip computing Inc., WINZIP 8.0, hereinafter WinZip, Java, and Fults et al. Patent #5327529, hereinafter Fults.

6. With regard to claim 1, WinZip teaches, on pages 3 and 4 a system of software components adapted to display text running under an operating system, in which selection of the mask password check box displays the text with one or two software components namely masked ("*****") or unmasked ("password"), the selection of which is made at runtime. Java teaches a system of masking passwords similar to that of WinZip using the Swing API and namely the JPasswordField and JTextField (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField), but further teaches a system independent display (see IS page 1 paragraphs 1 and 5). WinZip and Java, however don't teach using a peer component for selection between two proxy components. Fults teaches a system which generates the interface based on a selection by the user (see column 3, lines 27-52) similar to that of WinZip and Java, Fults, however, further teaches taking hints from a user and using the hints to direct the interface generation to an appropriate user interface implementation (see column 3, lines 33-51, figure 2, and the abstract). Fults further teaches, in column 20, lines 24-34, that libraries are dynamically loaded into memory when needed by the application. It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip and Java before him at the time the invention was made to modify the text display system of WinZip to include the system independence of Java. One would have been motivated to make such a combination

because this would allow people on different platforms to use similar and familiar interfaces. It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip, Java, and Fults, before him at the time the invention was made to modify the password display system of WinZip and Java to be implemented with a peer component that directs the interface development to the appropriate user interface implementation. One would have been motivated to make such a combination because WinZip has a switch of display characteristics very similar to that of Fults, where upon a user selection, an element is displayed using a different API component.

7. With regard to claims 2 and 14, which teach the object is part of a graphical user interface associated with the application program, WinZip teaches, on pages 3 and 4, that the object is part of a GUI associated with the application program WinZip.

8. With regard to claims 3 and 15, which teach the selection of the first or second of the systems of software components being made by the application, by sending an indication of the mode of use to the peer component, Fults teaches, in column 3, lines 33-51, figure 2, and the abstract, the indication of the specific user interface element being made to the peer component (the Generic User Interface Design).

9. With regard to claim 4 and 16, which teaches that the new graphics resource component is created during runtime, upon selection of the first or second proxy components to replace a graphics resource component previously created for the display object, Fults further teaches, in column 20, lines 24-34, that libraries are dynamically loaded into memory when needed by the application.

Art Unit: 2173

10. With regard to claims 5 and 17, which teach an instance of the first or second of the system of software components previously selected by the application program is destroyed when the instance is no longer selected, WinZip teaches, on pages 3 and 4, that selection of the software used to display the text is made in the application program WinZip through the use of the checkbox, and if there is existing text in the box when selection is made it will be destroyed and the new selected format will be displayed.

11. With regard to claims 6 and 18, which teach the appearance and behavior of the object differ depending on whether the object is displayed by the first or second of the system of software components (proxy components), WinZip further teaches, on page 3 and 4, that the appearance is different whether or not the mask is checked. WinZip also teaches, on page 6, that if the password is masked the user will be required to type it twice.

12. With regard to claims 7 and 19, which teach that the application program is written in Java programming language, WinZip and Fults teaches the system for displaying the text in two different formats, but doesn't teach the use of Java programming language. Java teaches a system of masking password similar to that of WinZip, but further teaches the use of Java programming language, specifically the Swing API and namely the JPasswordField and JTextField (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField). It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip, Fults, and Java before him at the time the invention was made to modify the text display system of WinZip and Fults to use Java. One would have been motivated to make such a

combination because this would allow people on different platforms to use similar and familiar interfaces.

13. With regard to claims 8 and 20, which teach the graphic resource components are included in a set of software components comprising the Swing application program interface (API), WinZip teaches the system for displaying the text in two different formats, but doesn't teach the use of Swing API. Java teaches a system of masking passwords similar to that of WinZip but further teaches the use of Java's Swing API and namely the JPasswordField and JTextField (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField). It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip and Java before him at the time the invention was made to modify the text display system of WinZip to use Java's Swing API. One would have been motivated to make such a combination because this would allow people on different platforms to use similar and familiar interfaces.

14. With regard to claims 9 and 22, which teach the operating system comprises Windows, Unix or OS/2 computer operating system, WinZip and Fults teaches the system for displaying the text in two different formats, but doesn't teach the operating system comprising Windows, Unix or OS/2 computer operating system. Java teaches a system of masking password similar to that of WinZip, but teaches the use of Java's Swing API which can be displayed with the same look and feel on Windows, Unix or Apple computers through the use of a components set code-named *Metal* (see *IS page 1, paragraphs 1- 5*). It would have been obvious to one of ordinary skill in the art,

Art Unit: 2173

having the teachings of WinZip, Fults, and Java before him at the time the invention was made to modify the text display system of WinZip to use Java's Swing API for platform independence. One would have been motivated to make such a combination because this would allow people on different platforms to use similar and familiar interfaces.

15. With regard to claims 10 and 21, which teach the first proxy component using JTextField and the second proxy component using the JPasswordField, of the Swing application program interface (API), WinZip and Fults teach the system for displaying the text in two different formats, but doesn't teach the use of a TextField and PasswordField. Java teaches a system of masking passwords similar to that of WinZip but teaches the use of Java's Swing API and namely the JPasswordField and JTextField (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField). It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip, Fults and Java before him at the time the invention was made to modify the text display system of WinZip and Fults to use Java's Swing API. One would have been motivated to make such a combination because this would allow people on different platforms to use similar and familiar interfaces.

16. With regard to claims 11 and 23, which teach the selection of either the first or second proxy components depends on the status of a software flag associated with the object, WinZip further teaches on page 3 and 4, that the appearance is different whether or not the mask is checked. Java Platform 1.2 Beta 4 API Specification: Class JPasswordField also teaches this in paragraph 1.

Art Unit: 2173

17. With regard to claims 12 and 24, which teach that the object is adapted to respond to text entry events and wherein the status of the software flag indicates whether or not a special character is echoed when text is entered, WinZip further teaches on page 3 and 4, that the appearance is different whether or not the mask is checked (characters are displayed if unselected, asterisks are displayed if selected). Java Platform 1.2 Beta 4 API Specification: Class JPasswordField also teaches this in paragraph 1 specifically mentioning an echoChar.

18. With regard to claim 13, WinZip teaches, on pages 3 and 4 a system of software components adapted to display text running under an operating system, in which selection of the mask password check box displays the text with one or two software components namely masked ("*****") or unmasked ("password"), the selection of which is made at runtime. WinZip further teaches typing in a character set with the first software component, monitoring the mask flag, and then changing the mode of use of the object upon selection of the flag with an appearance distinct from the first (see pages 3 and 4), and also that if the mask is selected the password must be typed twice (see page 6). Java teaches a system of masking passwords similar to that of WinZip using the Swing API and namely the JPasswordField and JTextField (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField), but further teaches a system independent display (see IS page 1 paragraphs 1 and 5). WinZip and Java, however don't teach using a peer component for selection between two proxy components. Fults teaches a system which generates the interface based on a selection by the user (see column 3, lines 27-52) similar to that of WinZip and Java,

Fults, however, further teaches taking hints from a user and using the hints to direct the interface generation to an appropriate user interface implementation (see column 3, lines 33-51, figure 2, and the abstract). Fults further teaches, in column 20, lines 24-34, that libraries are dynamically loaded into memory when needed by the application. It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip and Java before him at the time the invention was made to modify the text display system of WinZip to include the system independence of Java. One would have been motivated to make such a combination because this would allow people on different platforms to use similar and familiar interfaces. It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip, Java, and Fults, before him at the time the invention was made to modify the password display system of WinZip and Java to be implemented with a peer component that directs the interface development to the appropriate user interface implementation. One would have been motivated to make such a combination because WinZip has a switch of display characteristics very similar to that of Fults, where upon a user selection, an element is displayed using a different API component.

19. With regard to claim 25, WinZip teaches, on pages 3 and 4 a windows based operating system with a system of software components adapted to display text in an application program, in which selection of the mask password check box displays the text with one or two software components namely masked ("*****") or unmasked ("password"), the selection of which is made at runtime. WinZip further teaches typing in a character set with the first software component, monitoring the mask flag, and then

Art Unit: 2173

changing the mode of use of the object upon selection of the flag with an appearance distinct from the first (see pages 3 and 4), and also that if the mask is selected the password must be typed twice (see page 6). Java teaches a system of masking passwords similar to that of WinZip using the Swing API and namely the JPasswordField and JTextField (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField), but further teaches a system independent display (see IS page 1 paragraphs 1 and 5). WinZip and Java, however don't teach using a peer component for selection between two proxy components. Fults teaches a system which generates the interface based on a selection by the user (see column 3, lines 27-52) similar to that of WinZip and Java, Fults, however, further teaches taking hints from a user and using the hints to direct the interface generation to an appropriate user interface implementation (see column 3, lines 33-51, figure 2, and the abstract). Fults further teaches, in column 20, lines 24-34, that libraries are dynamically loaded into memory when needed by the application. It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip and Java before him at the time the invention was made to modify the text display system of WinZip to include the system independence of Java. One would have been motivated to make such a combination because this would allow people on different platforms to use similar and familiar interfaces. It would have been obvious to one of ordinary skill in the art, having the teachings of WinZip, Java, and Fults, before him at the time the invention was made to modify the password display system of WinZip and Java to be implemented with a peer component that directs the interface development to the appropriate user interface

implementation. One would have been motivated to make such a combination because WinZip has a switch of display characteristics very similar to that of Fults, where upon a user selection, an element is displayed using a different API component.

(10) Response to Argument

Claims 1-12:

With respect to the group of claims including Claims 1-12, the Appellant's arguments are focused on the limitations regarding the "existence of a first and second proxy component and a peer component, where the peer component is configured for selecting a proxy component for use". More specifically, as stated from representative Claim 1, the limitation argued is:

a peer component for selecting either the first proxy component or the second proxy component, depending on a mode of use of the object, wherein the selection can be made during runtime, and wherein after the proxy component is selected, the selected proxy component dynamically creates a new graphics resource component for displaying the object, such that the appearance of the displayed object is substantially independent of the operating system.

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The Examiner asserts the limitation is a

component used for selecting a display system used to display an object. As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

“Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure. In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997).”

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the references, are within the scope of these limitations.

-WinZip teaches, on pages 3 and 4 a system of software components adapted to display text running under an operating system, in which selection of the mask password check box displays the text with one or two software components namely masked (“*****”) or unmasked (“password”), the selection of which is made at runtime.

-Java teaches a system of masking passwords similar to that of WinZip using the Swing API and namely the JPasswordField and JTextField (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField), but further teaches a system independent display (see IS page 1 paragraphs 1 and 5).

-Fults teaches a system which generates the interface based on a selection by the user (see column 3, lines 27-52) similar to that of WinZip and Java, Fults, however, further teaches taking hints from a user and using the hints to direct the interface generation to an appropriate user interface implementation (see column 3, lines 33-51,

figure 2, and the abstract). Fults further teaches, in column 20, lines 24-34, that libraries are dynamically loaded into memory when needed by the application.

The examiner will now address the individual arguments and statements made by the Appellant.

From page of the Appeal Brief, from the fourth paragraph, the Appellant argues the screen shots on pages 3 and 4 of the WinZip reference cannot be used to provide teaching or suggestion for the presently claimed system of software components, which as noted above, include a first proxy component, a second proxy component and a peer component.

The examiner contends that the WinZip reference was included in the rejection to provide an illustration of how the display routine for a particular group of text can be changed during runtime to effect a change in the appearance of the displayed text (see pages 3 and 4). The Java reference is relied upon to teach the actual proxy components, JPasswordField and JTextField, which are display routines that are similar to those as in WinZip (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField). The Fults reference is further relied upon for the teaching of a peer component that selects between alternate user interface routines (see column 3, lines 33-51), similar to the selection of display routine by WinZip.

From page 8 of the Appeal Brief, from the fourth paragraph, the Appellant argues the WinZip reference does not teach or suggest that the functionality shown in the screen shots could be provided by software components similar to the presently claimed proxy and peer components.

The examiner contends that WinZip does show a display that implements two different display routines, that are obviously some sort of software component (see pages 3 and 4). Further there is a selection, by a user, two alternate between display routines (see pages 3 and 4).

From page 11 of the Appeal Brief, from the first paragraph, the Appellant argues the appearance of the displayed object would not be independent of the operating system, as in the presently claimed case, because a heavyweight graphic resource would be used for displaying the object in the WinZip GUI.

The examiner contends that a Java shows the display components are JPasswordField and JTextField, which are display routines that are similar to those as in WinZip (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField), which are Swing components, that have been shown by IS, page 2 to be able to be displayed with use of Metal which shows the same Look-and-Feel no matter what operating system it is implemented on.

From page 11 of the Appeal Brief, from the second paragraph, the Appellant argues the Java reference fails to provide teaching or suggestion for a

first proxy component, a second proxy component and a peer component for selecting between the first and the second proxy components, depending on the use of the object.

The examiner contends that the Java reference is relied upon to teach the actual proxy components, JPasswordField and JTextField, which are display routines that are similar to those as in WinZip (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField). The WinZip reference was included in the rejection to provide an illustration of how the display routines (further see Java reference) for a particular group of text can be changed (further see Fults) during runtime to effect a change in the appearance of the displayed text (see pages 3 and 4). The Fults reference is further relied upon for the teaching of a peer component that selects between alternate user interface routines (see column 3, lines 33-51), similar to the selection of display routine by WinZip.

From page 12 of the Appeal Brief, from the second paragraph, the Appellant argues the Java resource component may have combined features (e.g., masked and unmasked text capabilities) that simply do not exist within a single Swing graphics resource component.

The examiner contends that as can clearly be seen in the WinZip reference components that implement masked and unmasked text capabilities exist in the same resource component.

From page 12 of the Appeal Brief, from the third paragraph, the Appellant argues one skilled in the art would not consider the presently claimed proxy components to be an obvious feature in light of the WinZip and Java references.

In response to applicant's argument that the presently claimed proxy components to be an obvious feature in light of the WinZip and Java references, the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

The java reference provides a means of implementing the display routines of the WinZip reference which are dynamically changed at runtime.

From page 13 of the Appeal Brief, from the second and third paragraph, the Appellant argues Fults does not teach or suggest a peer component for displaying an object by selecting either a first or second proxy component, depending on a mode of use of the object" where "any selection between software components that may or may not be disclosed by Fults is not dependent on a mode of use of an object.

The examiner contends that Fults does show as admitted to by the applicant on page 11, paragraph 3, an application defining the generically a UI object to one or more specific UI objects depending on the specific UI chosen, where this is selection its self is

a selection of a mode of use. Further the selection of masked or unmasked by the WinZip reference is further a selection of a mode of use, requesting either visible text or hidden.

From page 14 of the Appeal Brief, from the fifth paragraph, the Appellant argues There is no motivation to combine or modify the teachings of the cited art to provide a proxy component configured for displaying an object by selecting, during runtime, either a first or a second proxy component, depending on a mode of use of the object.

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, WinZip teaches the broad idea of using two different display routines for an element, selectable by another component, and the Java and Fults references supplement implementations of the display routines and the selection component.

From page 16 of the Appeal Brief, from the fourth paragraph, the Appellant argues There is no motivation to combine the text display system of WinZip with the system independence of Java, as suggested by the Examiner.

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, WinZip teaches the broad idea of using two different display routines for an element, selectable by another component, and the Java and Fults references supplement implementations of the display routines and the selection component. Further the implementation of an idea in a different programming language is not novel.

From page 17 of the Appeal Brief, from the first paragraph, the Appellant argues The Examiner has failed to adequately support and/or establish prima facie grounds of obviousness.

In response to applicant's argument that there is no teaching or motivation to suggest the aforementioned limitations of present claim 1, the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly

Art Unit: 2173

suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art.

See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

Claims 13-25:

With respect to the group of claims including Claims 13-25, the Appellant's arguments are focused on the limitations regarding the "upon detection of a change in the mode of use of the object, deactivating the first proxy component and then activating a second proxy component to dynamically generate a second graphical representation of an object". More specifically, as stated from representative Claim 1, the limitation argued is:

Upon detecting a change in the mode of use of the object, deactivating the first component and activating a second proxy component to dynamically generate a second graphical representation of the object during runtime, wherein the second graphical representation is substantially independent of the operating system and distinct from the first graphical representation.

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The Examiner asserts the limitation is a selection that stops a first component from displaying in its manner and implements a second component to display the object. The mode of use is not limited in the way in

Art Unit: 2173

which it is changed, this can be an observed change or a user selected change. As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

“Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure. In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997).”

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the references, are within the scope of these limitations.

-WinZip teaches, on pages 3 and 4 a system of software components adapted to display text running under an operating system, in which selection of the mask password check box (user selection of a change in mode) displays the text with one or two software components namely masked (“*****”) or unmasked (“password”), the selection of which is made at runtime.

-Java teaches a system of masking passwords similar to that of WinZip using the Swing API and namely the JPasswordField and JTextField (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField), but further teaches a system independent display (see IS page 1 paragraphs 1 and 5).

-Fults teaches a system which generates the interface based on a selection by the user (user selection of a change in mode) (see column 3, lines 27-52) similar to that of WinZip and Java, Fults, however, further teaches taking hints from a user and using the hints to direct the interface generation to an appropriate user interface

implementation (see column 3, lines 33-51, figure 2, and the abstract). Fults further teaches, in column 20, lines 24-34, that libraries are dynamically loaded into memory when needed by the application.

The examiner will now address the individual arguments and statements made by the Appellant.

From page 19 of the Appeal Brief, from the first paragraph, the Appellant argues teachings of WinZip and Java references can combined to render the above limitations.

The examiner contends that the WinZip reference was included in the rejection to provide an illustration of how the display routine for a particular group of text can be changed during runtime to effect a change in the appearance of the displayed text (see pages 3 and 4). The Java reference is relied upon to teach the actual proxy components, JPasswordField and JTextField, which are display routines that are similar to those as in WinZip (see Java Platform 1.2 Beta 4 API Specification: Class JPasswordField and Class JTextField). The Fults reference is further relied upon for the teaching of a peer component that selects between alternate user interface routines (see column 3, lines 33-51), similar to the selection of display routine by WinZip.

From page 20 of the Appeal Brief, from the second paragraph, the Appellant argues the screen shots provide absolutely no teaching, suggestion or motivation for activating/deactivating different software components.

The examiner contends that the WinZip reference was included in the rejection to provide an illustration of how the display routine for a particular group of text can be changed during runtime to effect a change in the appearance of the displayed text (see pages 3 and 4). It is further the mode of use is not limited in the way in which it is changed, this can be an observed change or a user selected change.

From page 21 of the Appeal Brief, from the second paragraph, the Appellant argues The Examiner has failed to adequately support and/or establish prima facie grounds of obviousness.

In response to applicant's argument that there is no teaching or motivation to suggest the aforementioned limitations of present claim 13 and 25, the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference; nor is it that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981).

Art Unit: 2173

(11) Related Proceeding(s) Appendix

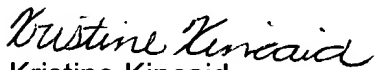
No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.


Respectfully submitted,



Dennis G. Bonshock
February 26, 2007



Kristine Kincaid
Supervisory Patent Examiner
February 26, 2007



Weilun Lo
Supervisory Patent Examiner
February 26, 2007

CONLEY ROSE, P.C.
P.O. BOX 684908
AUSTIN, TX 78768